Deep Learning Basics Lecture 1: Feedforward

Princeton University COS 495

Instructor: Yingyu Liang

Motivation I: representation learning

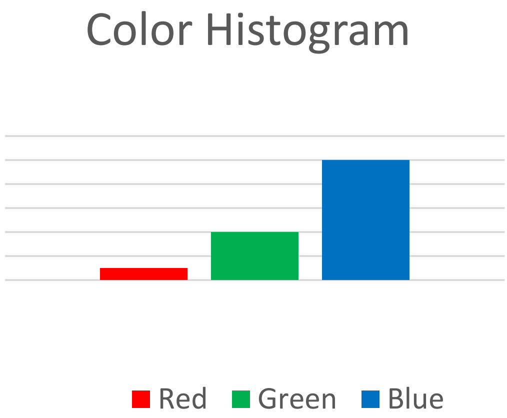
Machine learning 1-2-3

- Collect data and extract features
- Build model: choose hypothesis class $\mathcal{H}$ and loss function l
- Optimization: minimize the empirical loss

Features

x

Extract features

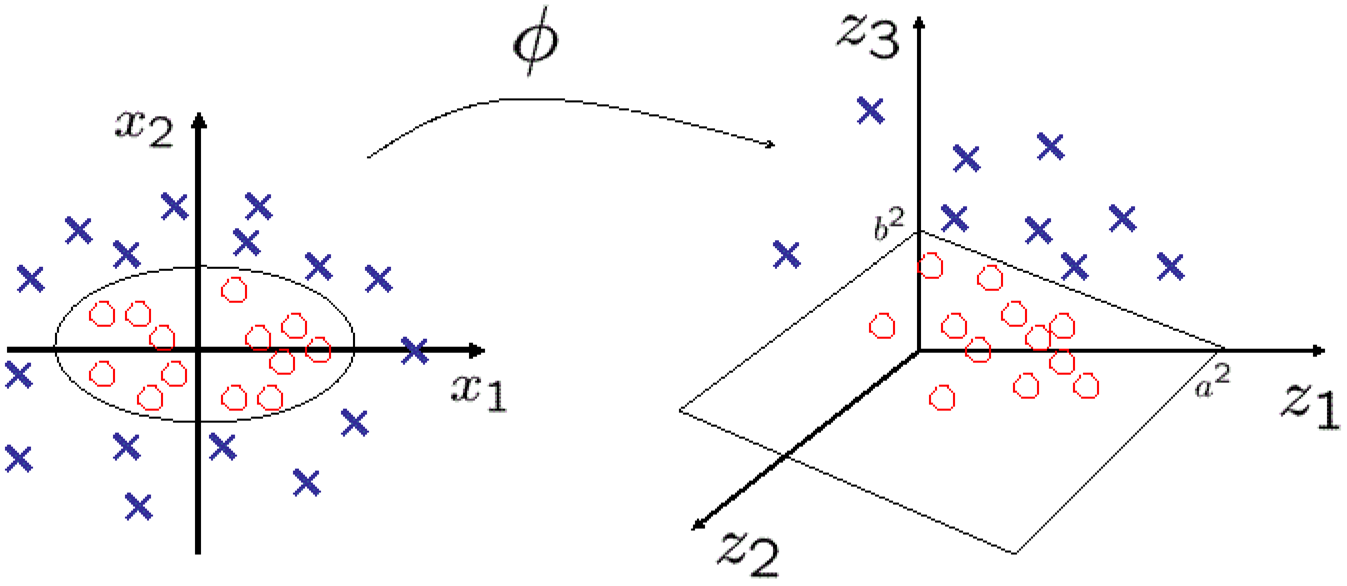


build hypothesis

$$y = w^T \phi(x)$$

Features: part of the model

Nonlinear model



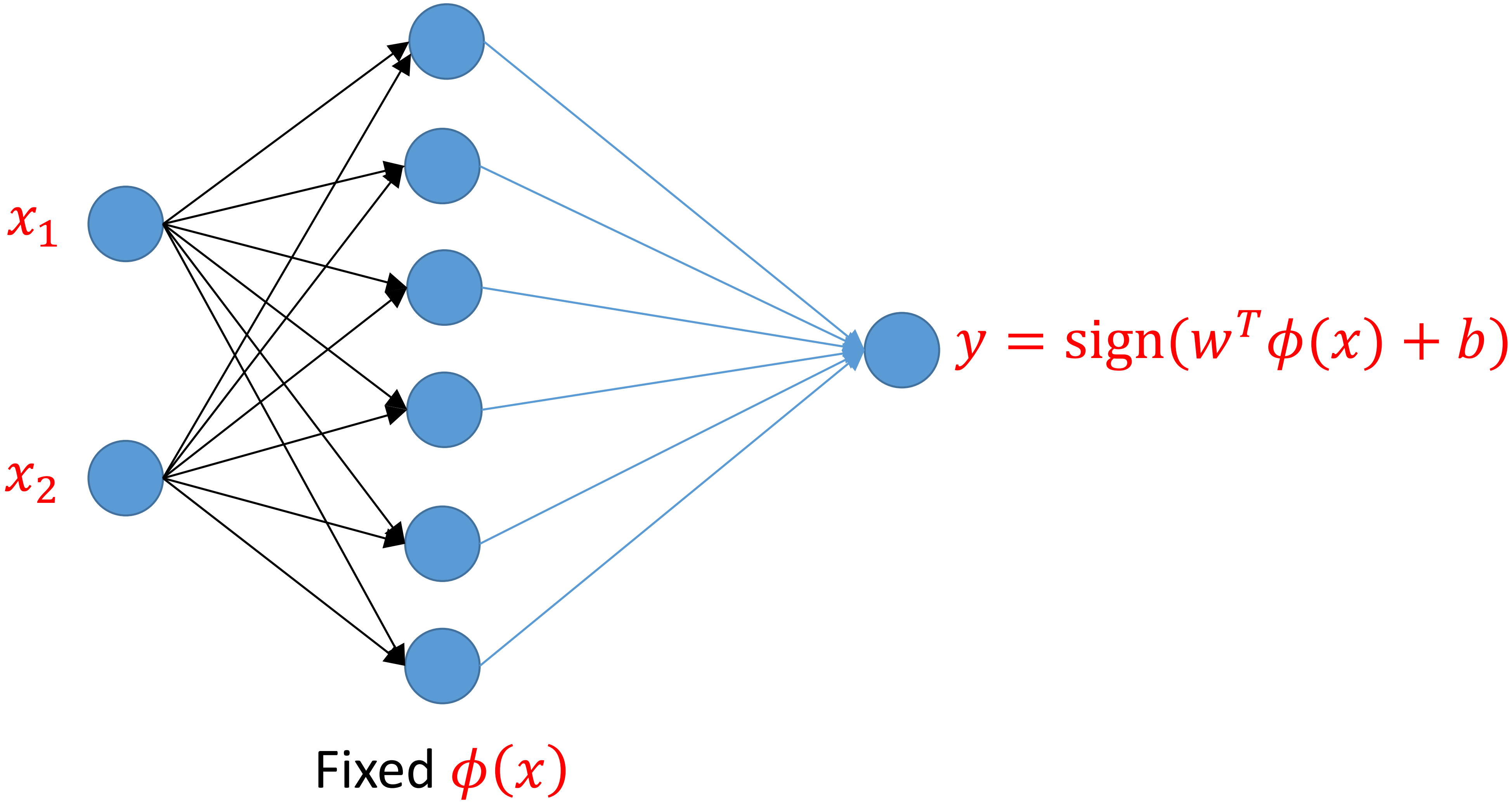
build

hypothesis

$y = w^T\phi(x)$

Linear model

Example: Polynomial kernel SVM



Motivation: representation learning

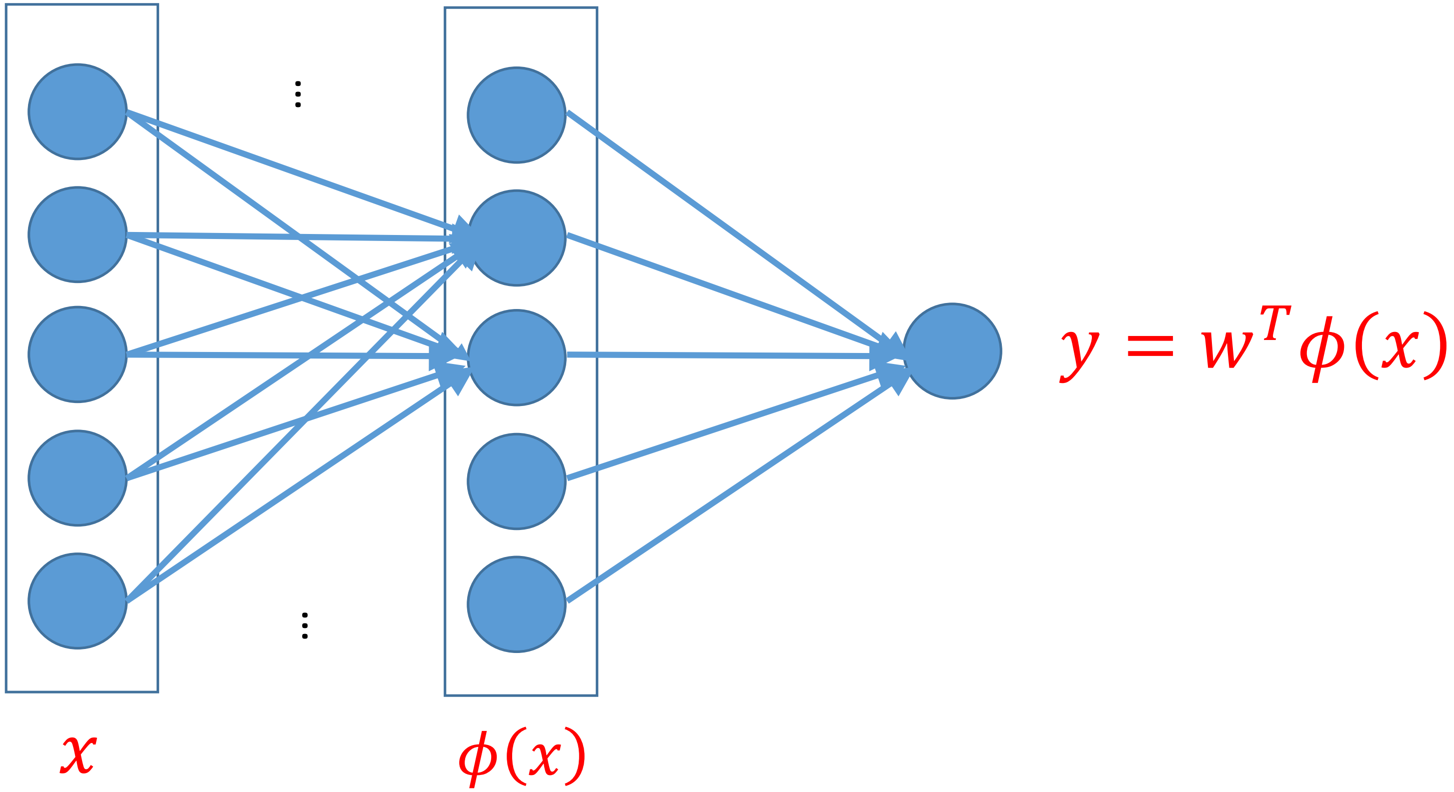
- Why don't we also learn $\phi(x)$?

x

Learn $\phi(x)$ $\longrightarrow$ $\boxed{\phi(x)}$ $\xrightarrow{\text{Learn } w}$ $y = w^T\phi(x)$

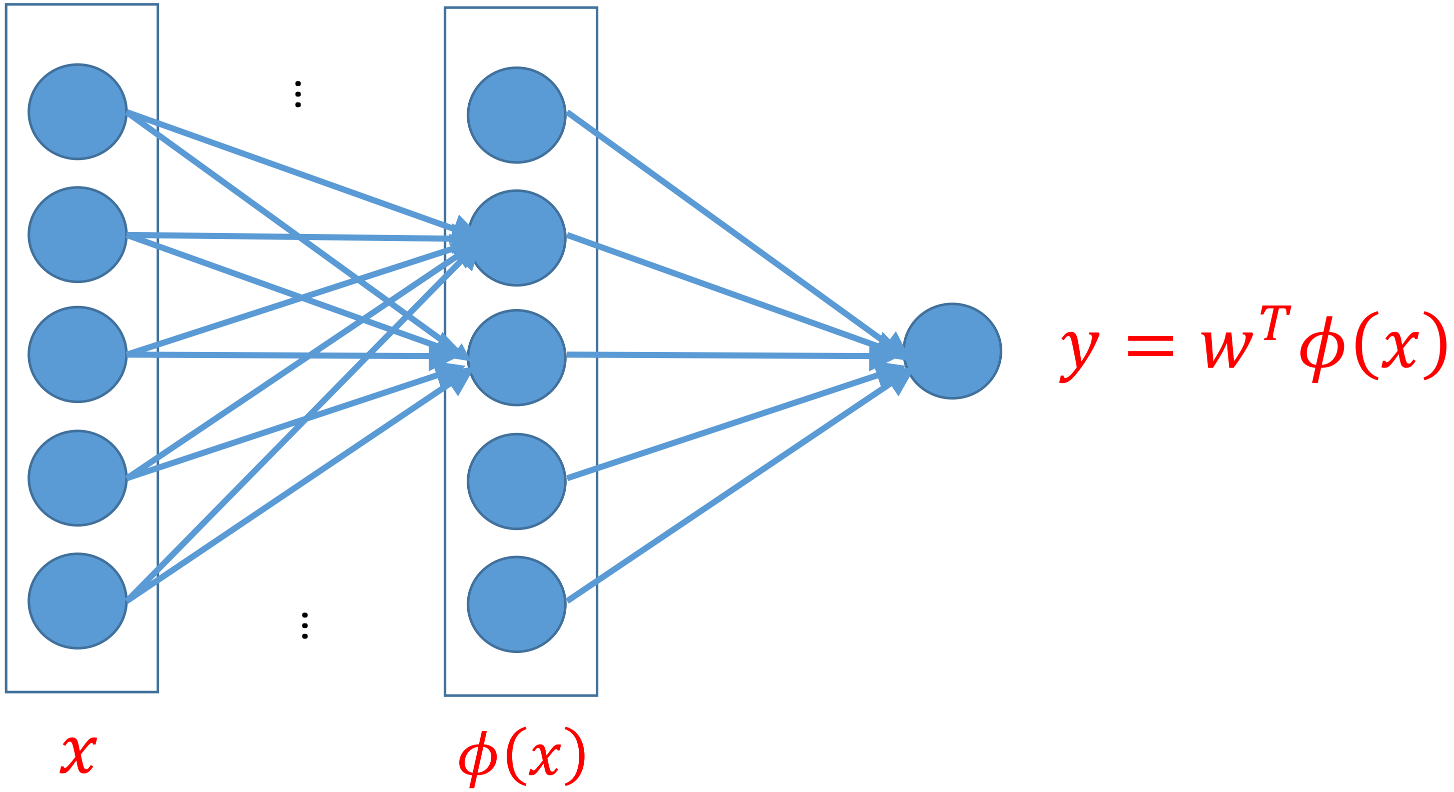
Feedforward networks

- View each dimension of $\phi(x)$ as something to be learned



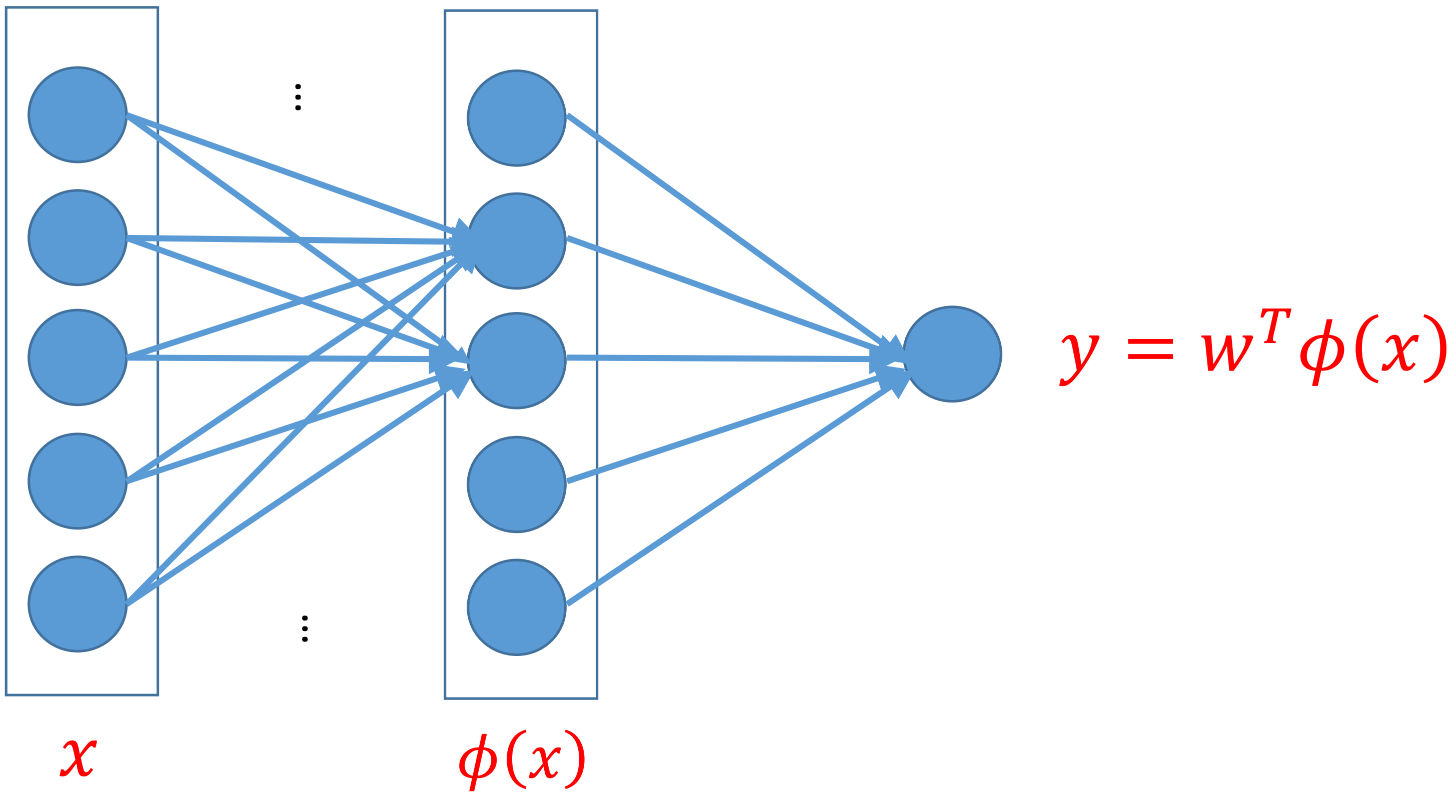
Feedforward networks

- Linear functions $\phi_i(x) = \theta_i^T x$ don't work: need some nonlinearity



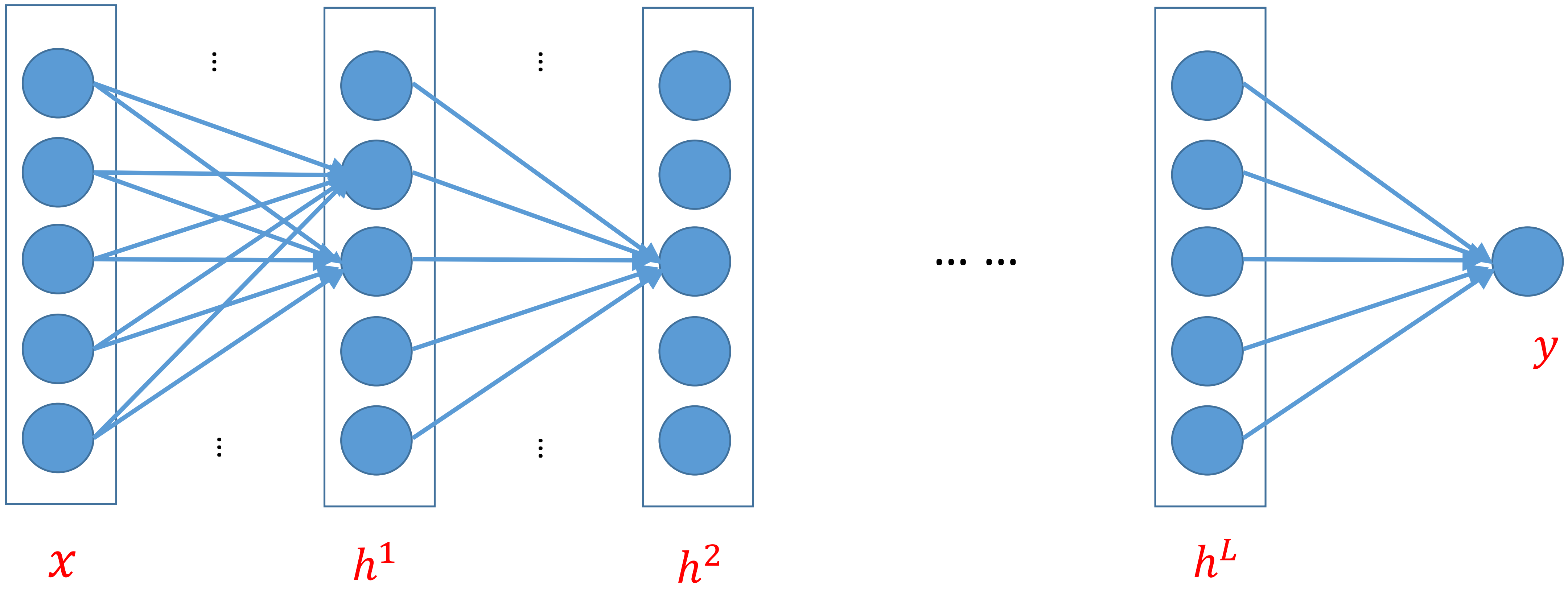
Feedforward networks

- Typically, set $\phi_i(x) = r(\theta_i^T x)$ where $r(\cdot)$ is some nonlinear function



Feedforward deep networks

- What if we go deeper?



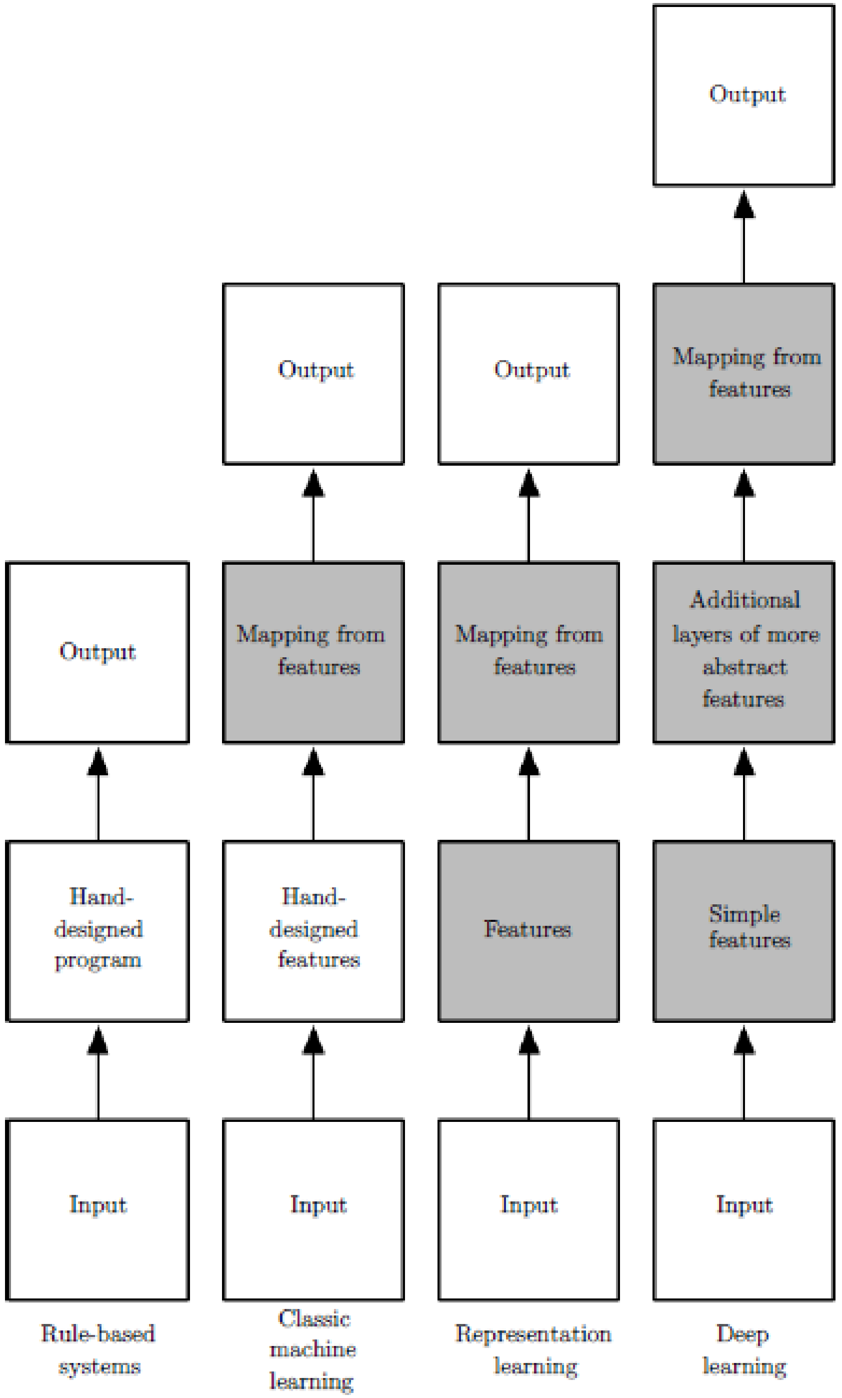


Figure from
Deep learning, by
Goodfellow, Bengio, Courville.
Dark boxes are things to be learned.

Motivation II: neurons

Motivation: neurons

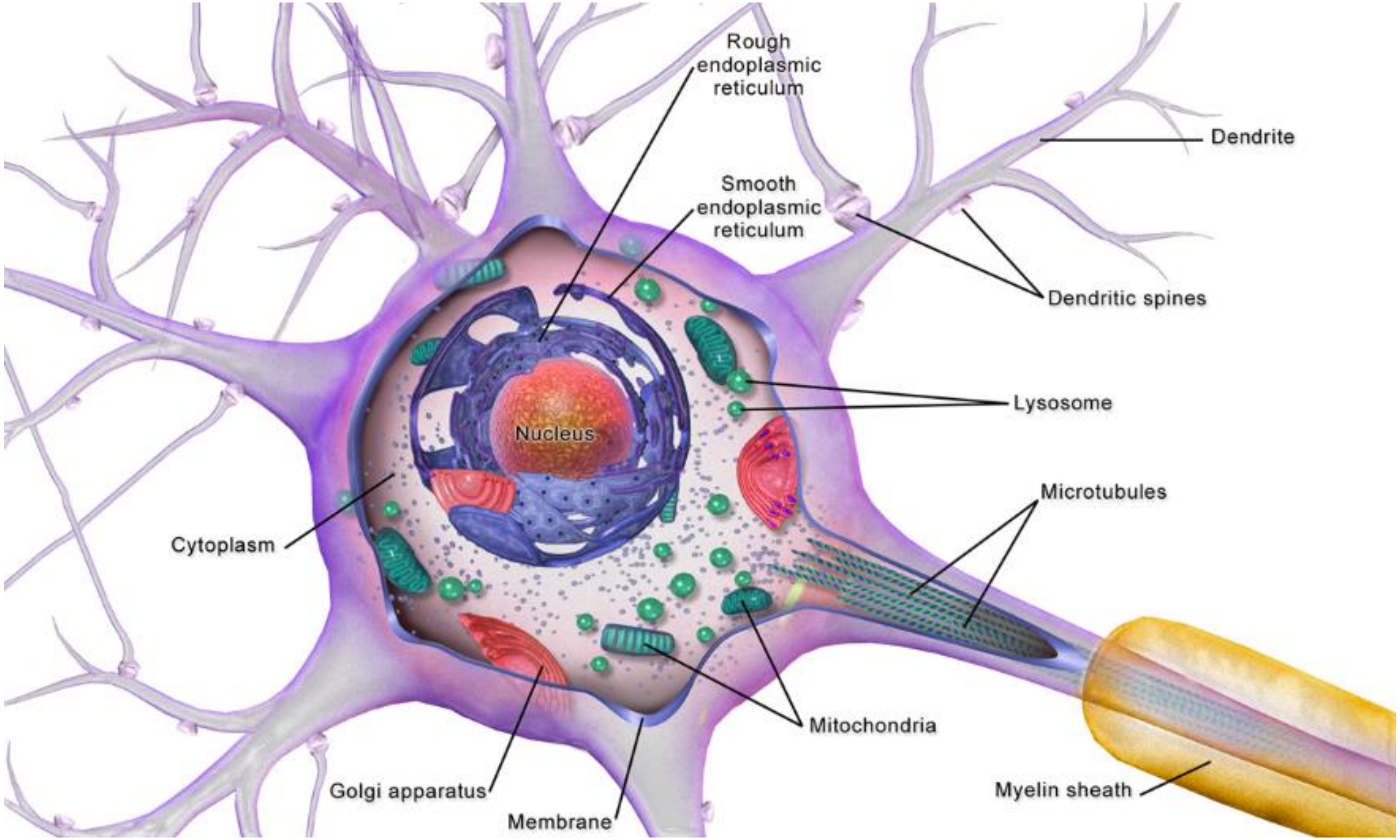
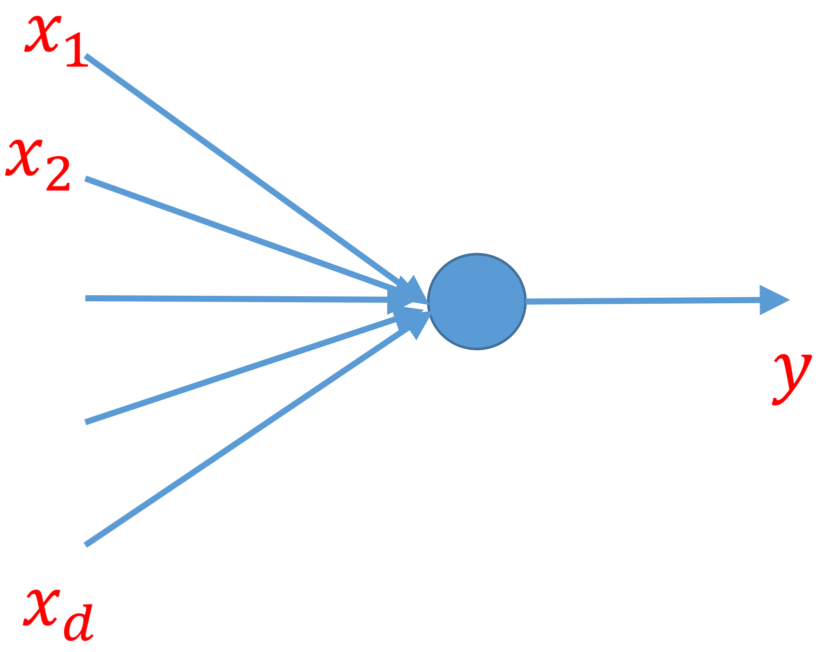


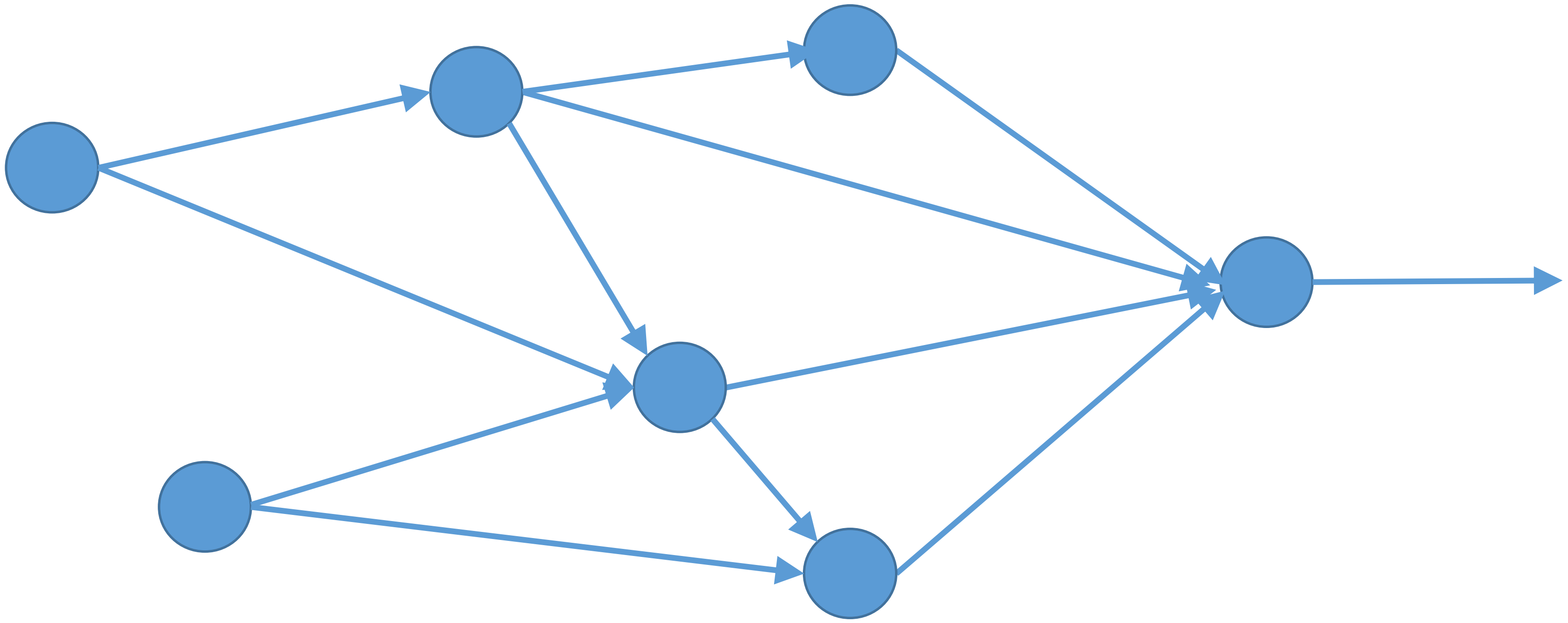
Figure from Wikipedia

Motivation: abstract neuron model

- Neuron activated when the correlation between the input and a pattern θ exceeds some threshold b
- $y = \text{threshold}(\theta^T x - b)$
  or $y = r(\theta^T x - b)$
- $r(\cdot)$ called activation function

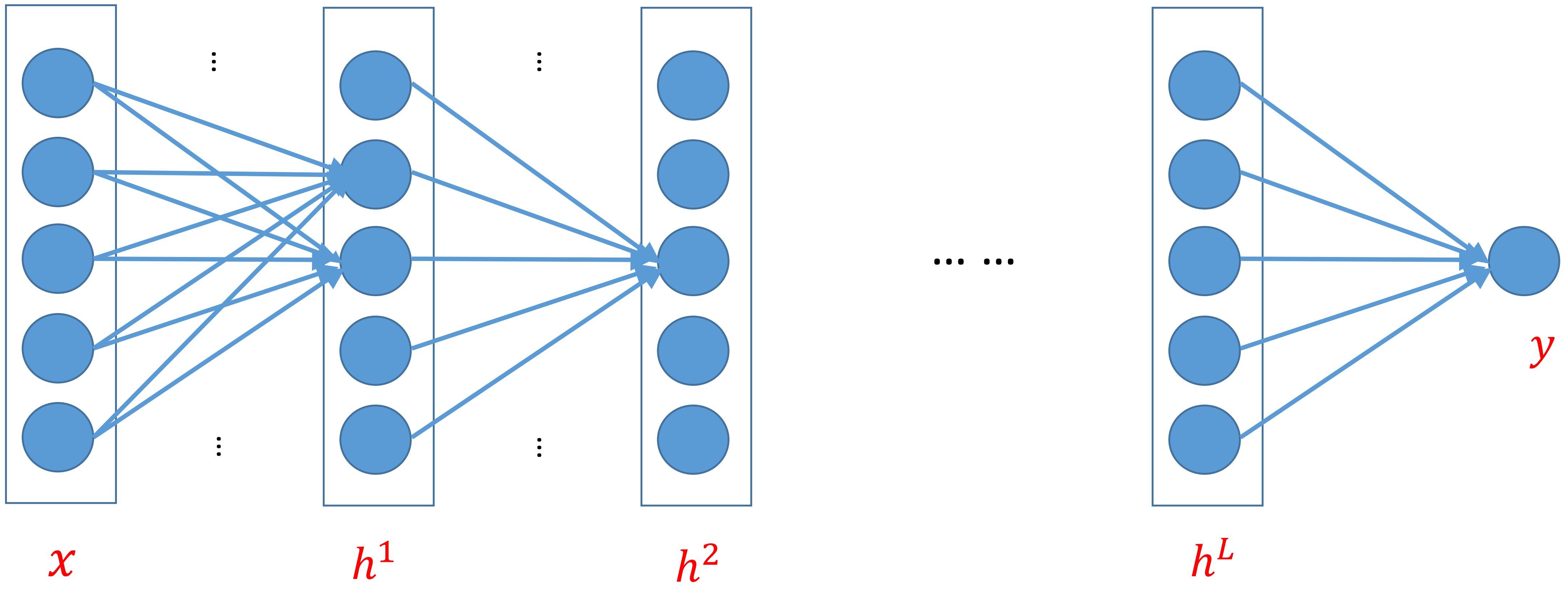


Motivation: artificial neural networks



Motivation: artificial neural networks

- Put into layers: feedforward deep networks

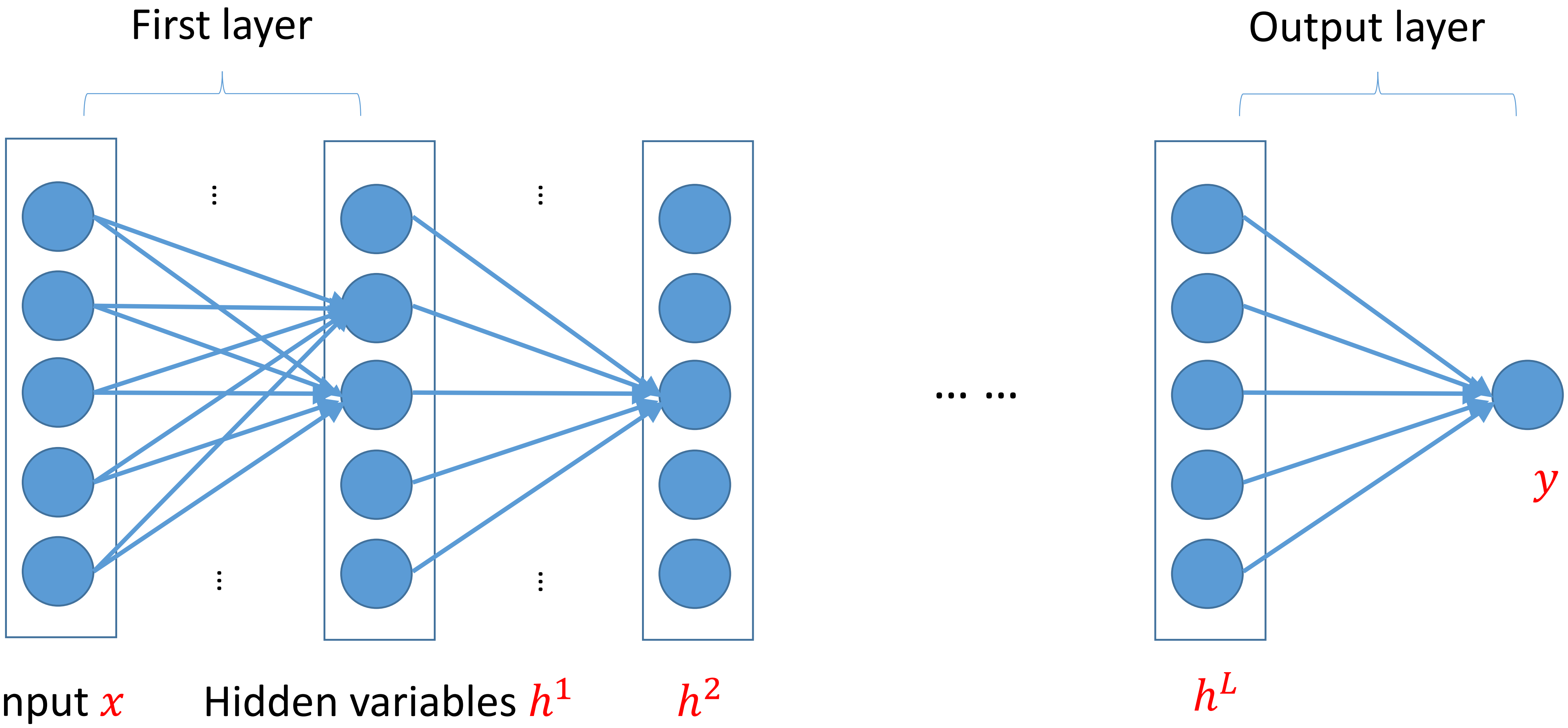


Components in Feedforward networks

Components

- Representations:
  - Input
  - Hidden variables
- Layers/weights:
  - Hidden layers
  - Output layer

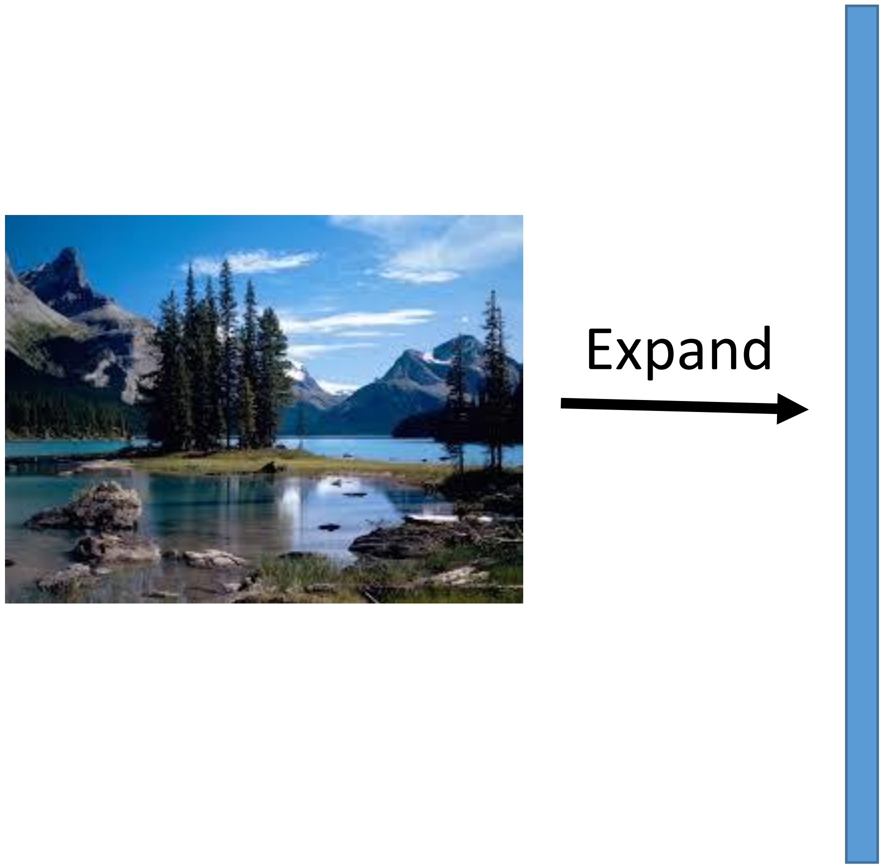
Components



Input

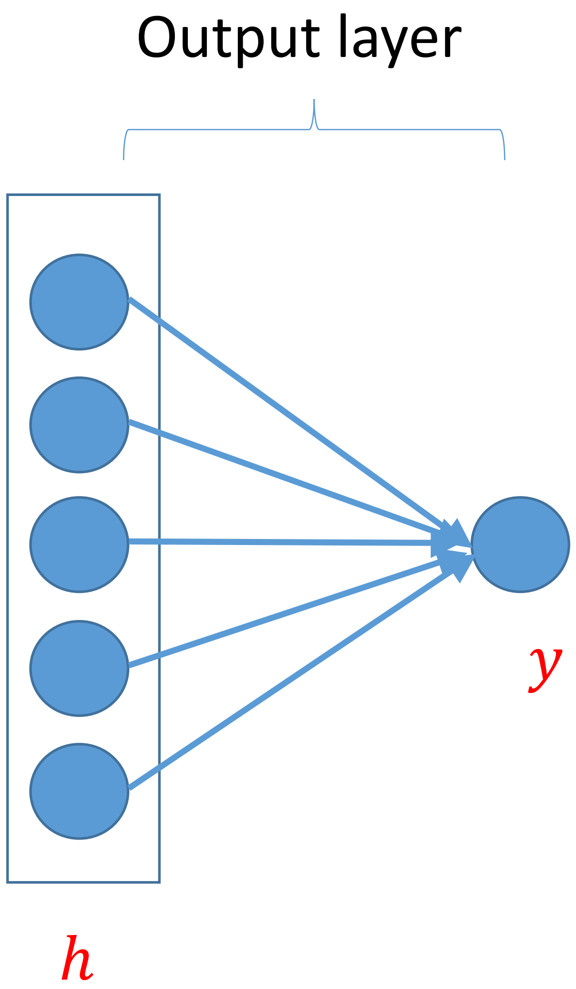
- Represented as a vector

- Sometimes require some preprocessing, e.g.,
  - Subtract mean
  - Normalize to [-1,1]



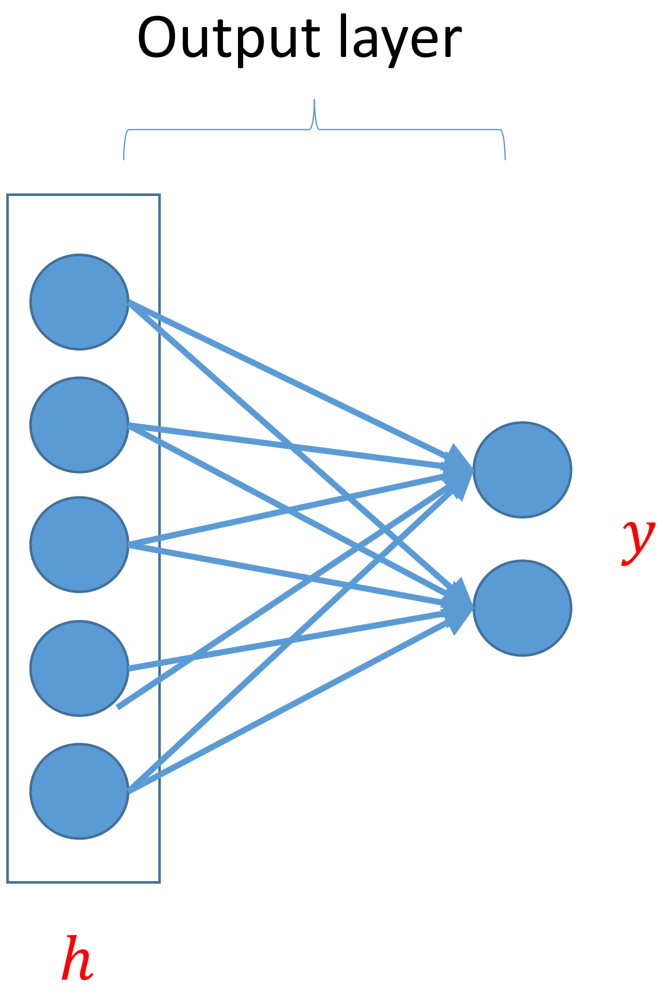
Output layers

- Regression: $y = w^T h + b$
- Linear units: no nonlinearity



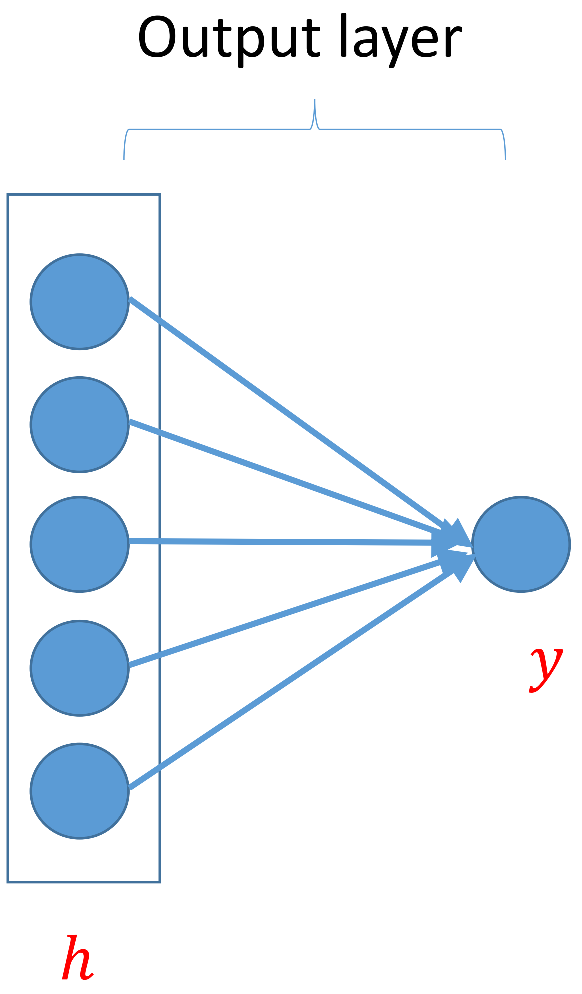
Output layers

- Multi-dimensional regression: $y = W^T h + b$
- Linear units: no nonlinearity



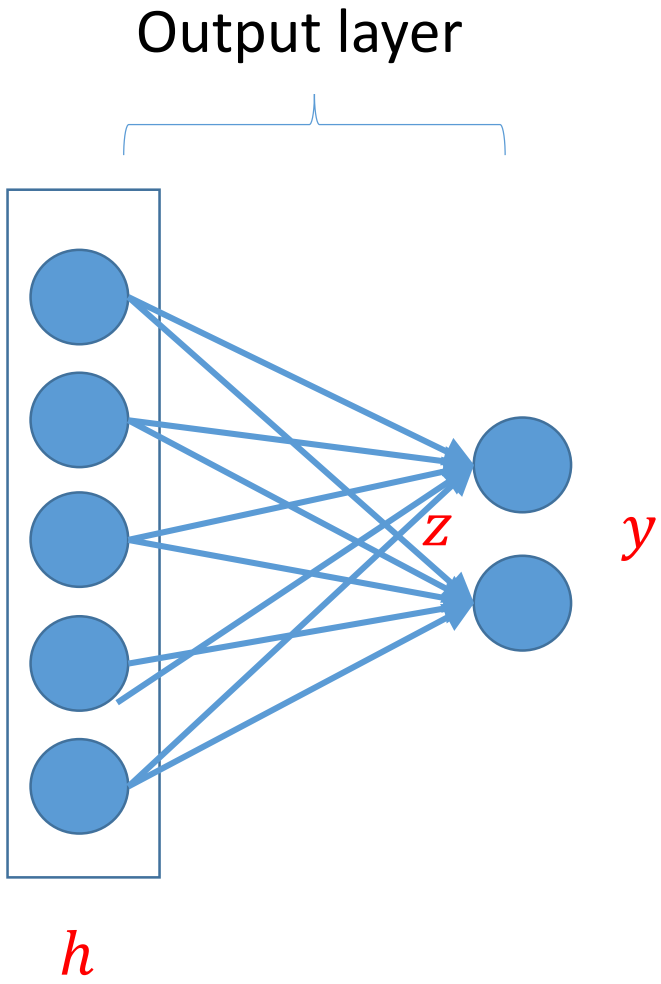
Output layers

- Binary classification: $y = \sigma(w^T h + b)$
- Corresponds to using logistic regression on h



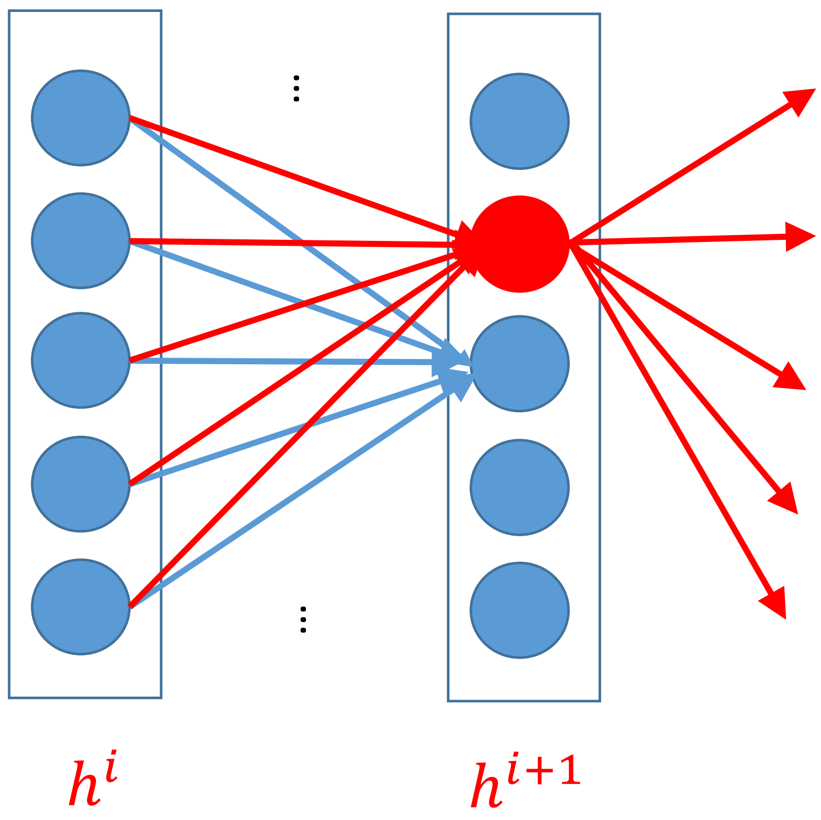
Output layers

- Multi-class classification:
- $y = \text{softmax}(z)$ where $z = W^T h + b$
- Corresponds to using multi-class logistic regression on h



Hidden layers

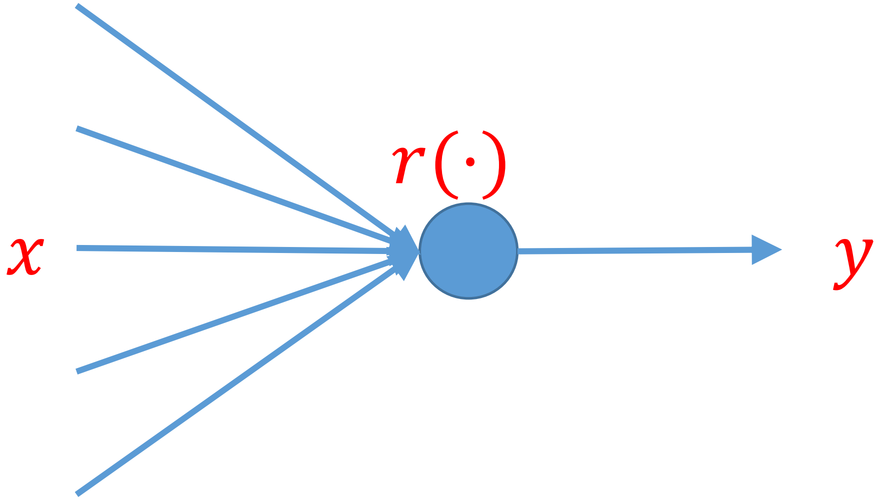
- Neuron take weighted linear combination of the previous layer
- So can think of outputting one value for the next layer



Hidden layers

- $y = r(w^T x + b)$

- Typical activation function r
  - Threshold $\mathrm{t}(z) = \mathbb{I}[z \geq 0]$
  - Sigmoid $\sigma(z) = 1/(1 + \exp(-z))$
  - Tanh $\tanh(z) = 2\sigma(2z) - 1$



Hidden layers

- Problem: saturation

Too small gradient

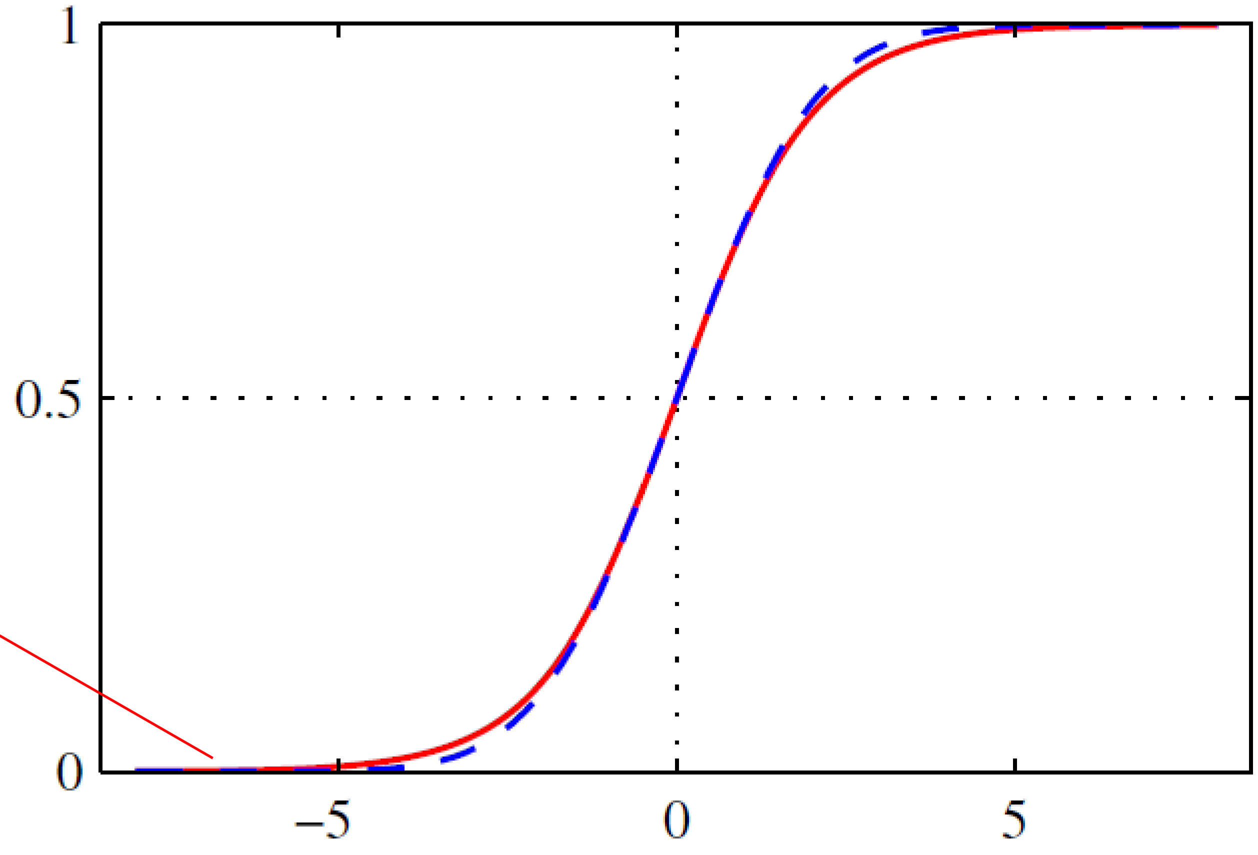


Figure borrowed from *Pattern Recognition and Machine Learning*, Bishop

Hidden layers

- Activation function ReLU (rectified linear unit)
  - $\mathrm{ReLU}(z) = \max\{z, 0\}$

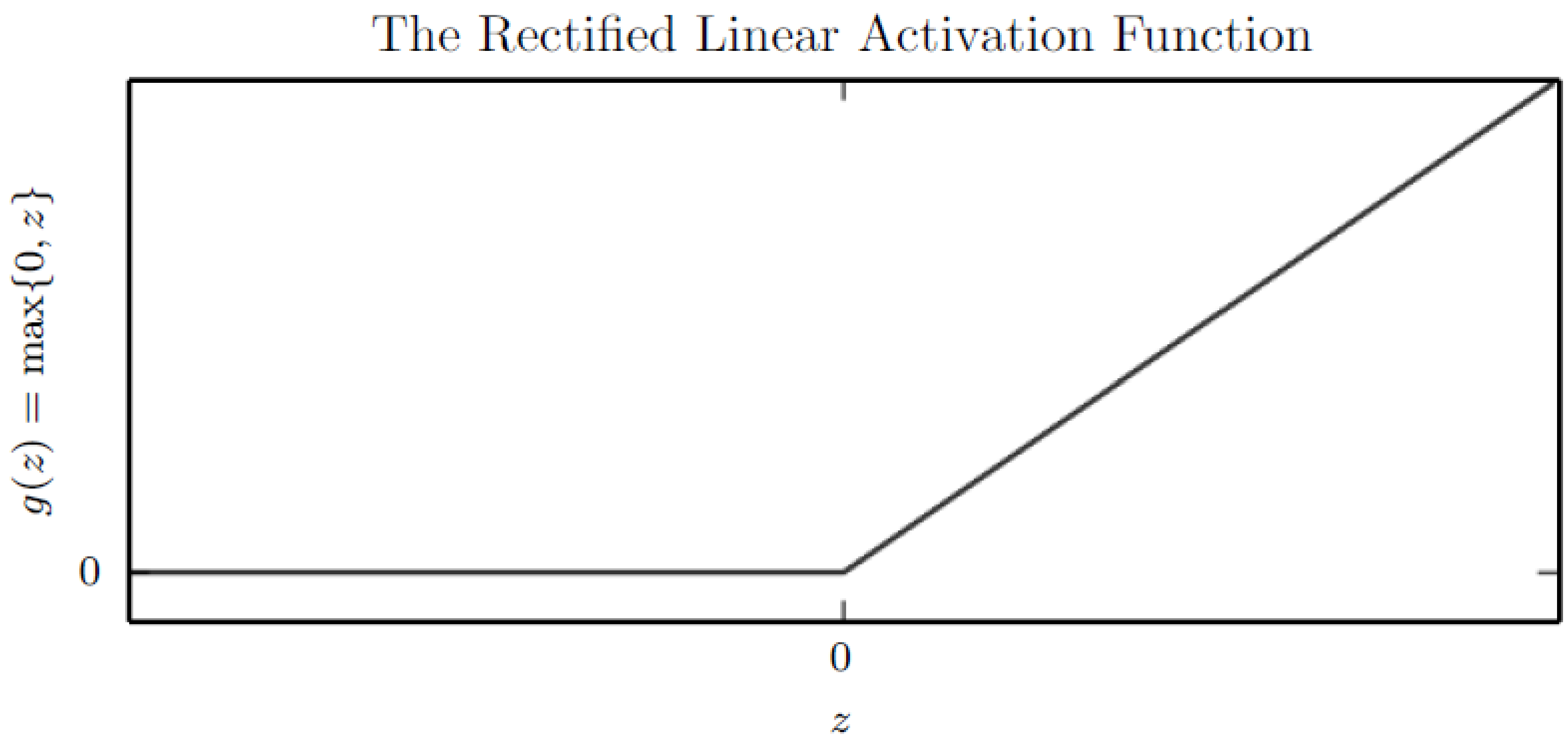
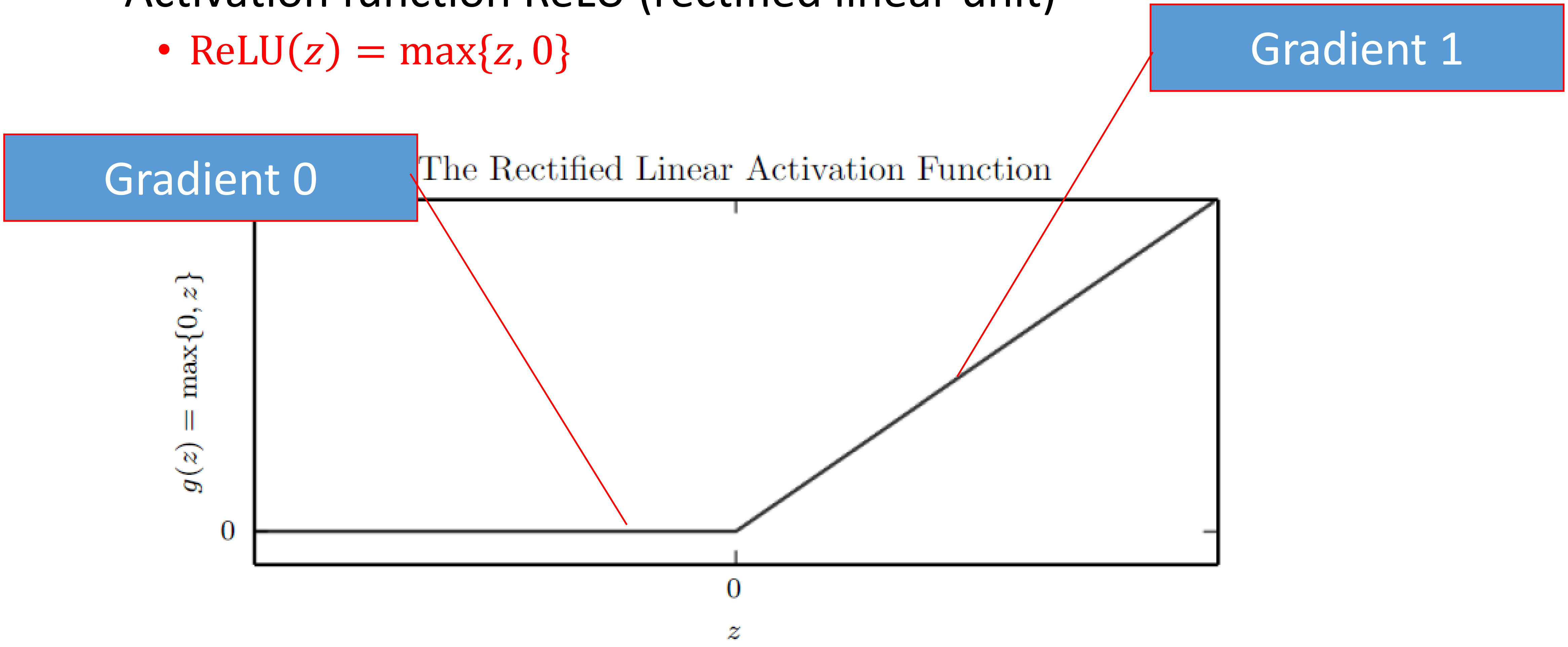


Figure from *Deep learning*, by Goodfellow, Bengio, Courville.

Hidden layers

- Activation function ReLU (rectified linear unit)
  - $\mathrm{ReLU}(z) = \max\{z, 0\}$



Hidden layers

- Generalizations of ReLU $\mathrm{gReLU}(z) = \max\{z, 0\} + \alpha \min\{z, 0\}$
  - $\text{Leaky-ReLU}(z) = \max\{z, 0\} + 0.01 \min\{z, 0\}$
  - $\text{Parametric-ReLU}(z)$: α learnable